Distributed Visualization and Virtual Environments for Meteorology and Oceanography

Robert Moorhead
Engineering Research Center
Mississippi State University
Miss. State, MS 39762
phone: (601) 325-2850 fax: (601) 325-7692 email: rjm@erc.msstate.edu
Award #: N00014-97-1-0116
http://www.erc.msstate.edu/thrusts/scivi/COST/cost/doc/index.html

LONG-TERM GOAL

My long term goal is to contribute to our understanding of the dynamics of METOC and ocean-atmosphere interactions. Of particular interest to me are analytical visualization systems which can show METOC models and measured data simultaneously so that a user can explore the complex physical dynamics.

OBJECTIVES

My objective is to develop a distributed visualization and virtual environment for METOC. This system would have readers for many ocean model outputs (e.g., NLOM, POM, WAM, etc.), for many meteorology model outputs (COAMPS, NOGAPS/NORAPS, etc.), and for measured data (CTDs, satellite data, scattered data, etc.).

APPROACH

We realized early on that we needed to prioritize our efforts. We decided to attack distributed visualization first, then concurrent visualization, and finally a virtual environment. To do such we decided we needed to build a visualization system that would allow us to create batch mode (off-screen) rendering packages, interactive (on-screen) visualization packages, and virtual environments with the same core functionality (color mapping, slicing planes, particle traces, isosurfaces, etc.).

Early in our effort we were told by our collaborators in the NRL Ocean Dynamics and Prediction Branch (NRL 7320) at Stennis Space Center, MS that what they most needed was a batch-mode, off-screen rendering toolkit that would let them make movies overnight of their daily model runs utilizing the parallel computation resources in the NAVO/MSRC. This became a significant focus.

WORK COMPLETED

We have developed a series of programs to allow users to do off-screen and on-screen rendering of terascale (6000 x 3000 x 6) time-varying datasets on desktop workstations. We have gone through a series of iterations with various weightings on the importance of:

- executing on a Sun versus just being able to display output on a Sun while the program actually runs remotely on an SGI machine,

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1998		**00-00-1998 to 00-00-1998**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Distributed Visualization and Virtual Environments for Meteorology and Oceanography	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **MIssissippi State University,Engineering Research Center,Mississippi State,MS,39762**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM002252.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**4**	

- 2D versus 3D renderings, and
- on-screen (interactive) versus off-screen (batch-mode) rendering.

The problem was trying to create an experiment for which we could obtain enough participation to clearly state the value of 3D renderings (views other than top or front) and the value of exploiting parallel computational resources. Existing programs (e.g., GMT) are capable of generating 2D renderings, but do not exploit parallel computational resources to speed-up the image generation process. We toiled for about 9 months trying to generate one program (evMovieLoop2D) which still rendered in 2D but exploited parallel resources for speed-up, one program (evInteractive2D) which rendered interactively in 2D and exploited parallel resources, one program (evInteractive3D) which rendered 3D surfaces off-screen and exploited parallel resources, and one program (evInteractive3D) which rendered 3D surfaces interactively and exploited parallel computational resources, hopefully to make the image generation fast enough to warrant interactive use. We were forced to abandon this four-pronged effort for lack of sufficient funding and personnel time and to create one program to do all four tasks. The desire to do something useful exceeded our desire to perform curiosity research.

We have converged to a single executable which uses MesaGL for off-screen rendering and SGI OpenGL for interactive rendering. A pre-processing program tiles the data into 4D tiles. The user determines the dimensions of the tiles. This allows the user to exploit parallel computers and to move into memory only the geographical or temporal data he or she wants. The program is built on a hierarchy of C++ libraries called COST (see the website cited).

Our requirements for this architecture were that it:

- be open and extensible
  - We hope for the system to grow large, but not monolithic.
  - Contributed modules should work just like core modules.
  - Only minimal knowledge should be required to extend modules.
- use OpenGL
  - For simplicity we use a scene-graph API for much rendering.
  - This allows for support of interactive and immersive vis in the future.
- present scientists with a unified interface style
  - Because tools draw from a shared code base, capabilities are similar and interface parameters are similar even if the context of their use is drastically different.
  - This does not mean that every program must adhere to some rigid specification, but an effort to provide continuity reduces accidental complexity.
- allow for a wide range of functionality
  - The same objects should be usable in applications that write out movie loops in batch style, run in an interactive window, or in a virtual environment.
  - This means that effort to enhance 'shared' objects are transparently available to the applications that use them.
- encourage use of multiple processors
  - The trick is to do this without binding unnecessarily to a particular thread or message passing library.

    - Encourage by providing objects that can make use of multiple threads (through a thread pool interface, for example).
- allow for scripting
    - Perl has been chosen as the scripting language because it is a well-supported, well-known language (minimize the needlessly arcane).
    - Scripts can be allowed to access and modify the program as if they were compiled into it.
- be compatible with emerging standards
    - EnVis objects are built primarily on ANSI/ISO C++.
    - Owing to the multi-executable nature of the EnVis architecture, extending into use of CORBA, virtual environments, distributed visualization, etc. will not be hampered by the need to get everything to compile into one application.

RESULTS

Our original idea was to have 5 executables (programs, toolkits): evMovieLoop2D, evMovieLoop3D, evInteractive2D, and evInteractive3D, and evVirtualEnv. We wanted to do this to exploit the inherent speed and ease in only rendering 2D and to allow the batch executables to run on more platforms. We learned this porfolio was going to be very difficult to develop and maintain. We also found a way to make one executable work efficiently for the first four needs by using dynamic libraries. We have yet to make significant progress on the virtual environment (VE) version due to much greater interest in a better batch and interactive system by our ocean modeling colleagues/collaborators and a lack of their interest in a VE system due in part to the lack of available VE hardware. However, we have extended cthru, our existing oceanographic virtual environment system to handle a dataset from the Chesapeake Bay and will be showing that at SC98 in collaboration with personnel from NRL/SSC.

We have spent months getting all the necessary contextual information to automatically appear in the image – lat/long legends, date of timestep, model run number, etc. – in a suitable presentation fashion. For example, we were asked to get lat/long axes to start at integer multiples of 5 automatically based on the model extent, even if the model doesn't start at a nice value.

We have not yet created a parallel, distributed visualization system with all the functionality and capability of something like GMT. To do such would or will require much more effort.

Movies of model runs can now be created faster using either the batch or interactive model. For example, the existing workhorse program is called ISTV. ISTV can render 120 timesteps of a 192x224 dataset in 2:50 minutes to a 950x950 pixel window interactively on an Onyx2. EnVis interactively is about as fast. A set of pictures is appended showing how the tiled data is rendered in interactive mode.

EnVis can off-screen render the same dataset to the same size window using a single processor in 4:38 minutes. Using 2 processors, it takes only 2:28. Using 4 processors it can render it in 1:20 minutes and with 6 processors it can render it in 0:58 seconds. Note this is near linear speedup and can be run from any machine; no X-server is necessary.

IMPACT/APPLICATION

Movies of model runs can now be created faster using either the batch or interactive model due to the parallel processing capability. The Ocean Dynamics and Prediction Branch at NRL/SSC has an 8 processor Onyx2 in their lab. The Onyx2 is part of the NAVO/MSRC and thus on the internal highspeed networks. EnVis allows them to interactively generate movie loops faster by a factor of 8 using the Onyx2. Using the Origin 2000 in the MSRC, for example, and the off-screen rendering capability they can obtain generate movies even faster (linear speedup).

TRANSITIONS

The programs have been installed and are being installed in the NAVO/MSRC for the use of the NRL Ocean Dynamics and Prediction Branch, as well as other people interested in visualizing oceanographic datasets.

RELATED PROJECTS

1 – We are continuing to advance ISTV. We have improved the NLOM 3.4T reader, added a reader for WAM data, ported ISTV to the Sun platform, started adding a scripting capability, and fixed a number of problems. The primary customer is the NRL Ocean Dynamics and Prediction Branch, but personnel in the CEWES/MSRC are beginning to use it too.

2 – We have used cthru to allow a VE fly-thru of a dataset from the Chesapeake Bay that Paul Martin of the NRL Ocean Dynamics and Prediction Branch has generated. It will be shown at SC98 Nov. 9-12.

REFERENCES

PUBLICATIONS

C.W. Everitt, J. van der Zwaag, and R.J. Moorhead, • COST: Common Object Support Toolkit," C++ Toolbox, ACM SIGPLAN Notes, Feb. 1998.